Experiment No: 10 **Date:**

Title: Generation of PN Sequence

Objective: To write a MATLAB program to generate PN sequence using linear feedback shift register (LFSR), given the number of flipflops and the generator polynomial.

Theory:

Pseudo-Noise (PN) sequences are usually used in order to generate noise that is approximately "white". It has applications in scrambling, cryptography, and spread-spectrum communications. It is also commonly referred to as the Pseudo-Random Binary Sequence (PRBS). These are very widely used in communication standards these days. The qualifier "pseudo" implies that the sequence is not truly random. Actually, it is periodic with a (possibly large) period and exhibits some characteristics of a random white sequence within that period. PN sequences are generated by Linear Feedback Shift Registers (LFSR), as shown in the following Fig. 1:

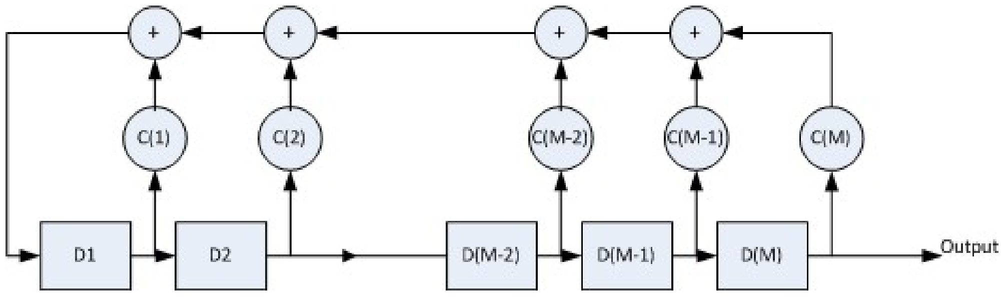


Fig. 1. General architecture of Linear feedback Shift Register

A PN data sequence is an M-sequence that is generated using a linear feedback shift-register circuit, as illustrated above. ***M*** is the number of shift registers. ***D(M)*** is the m^th^ shift register, and $\{\boldsymbol{c_1}, \boldsymbol{c_2}, \ldots, \boldsymbol{c_M}\}$ are the coefficients of them. At each clock pulse, the data in the registers will right shift once and one PN datum is output from register D(M).

Mathematically, the procedure can be defined by a generator polynomial. $\{\boldsymbol{c_1}, \boldsymbol{c_2}, \ldots, \boldsymbol{c_M}\}$ becomes the coefficients of the generator polynomial. For instance, the polynomial for PN9 is x^9+x^5+1, therefore, M=9 and

$$c_i = \begin{cases} 1, for\ i = 9\ and\ i = 5 \\ 0, for\ i \in \{1 \sim 9\}\ and\ i \neq 9, i \neq 5 \end{cases}$$

The PN sequence, $\{a_n\}$, $0 \leq n < 2^M$, is generated by the equation below.

$$a_n = \begin{cases} initial\ state\ in\ D^{M-n}, & 0 \leq n < M \\ c_M a_{n-M} + c_{M-1} a_{n-(M-1)} + c_{M-2} a_{n-(M-2)} + \cdots + c_2 a_{n-2} + c_1 a_{n-1}, & M \leq n < 2^M \end{cases}$$

where, the initial state of registers $\{D^1, D^2 \ldots D^M\}$ is the seed. D(M) stores the LSB of the seed, and D1 stores the MSB of the seed. For example, if the seed is 10 (binary form 1010), the initial state in register *{D¹,D²…Dᴹ}* is {0 0 0 0 0 1 0 1 0}. The Fig. 2 below shows the initial state of each register for PN9. It's obvious that LSB of the seed comes out first.

Afterwards, the registers store previously generated data, $D^m = a_{n-m}, m \in (1, M)$.

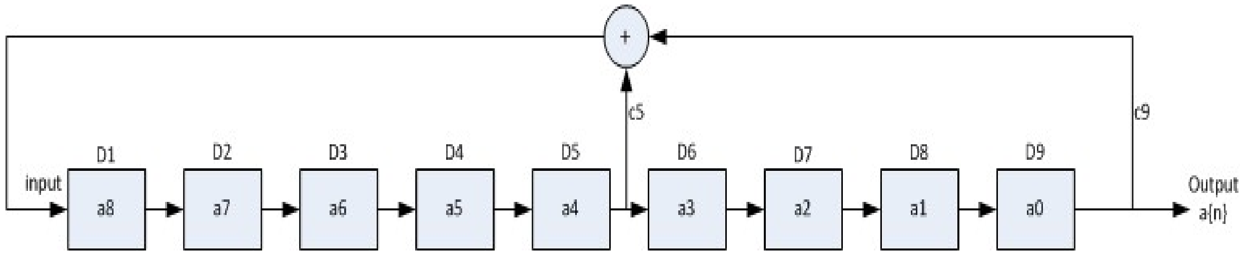


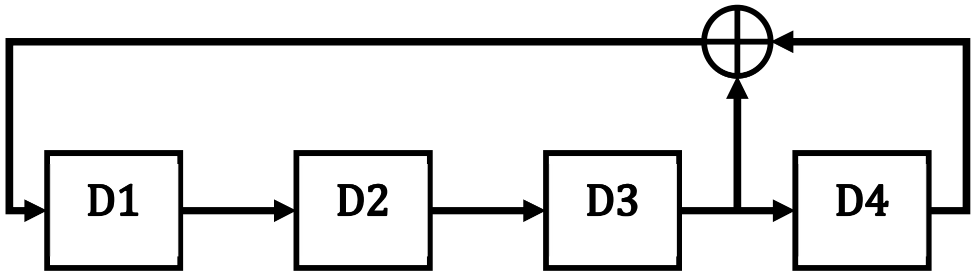
Fig. 2 LFSR for PN9

Design of PN sequence generator:

Given the generator polynomial $x^4 + x^3 + 1$,

Here M = 4 the period of the PN sequence $N = 2^M - 1 = 16 - 1 = 15$

Assuming the initial state {1,0,0,0}



Initial State 1 0 0 0

Fig. 3 LFSR architecture for PN4 for generator polynomial $x^4 + x^3 + 1$

The following table illustrates the operation of the LFSR for PN4 generator with respect to each clock pulse.

Clock	D1	D2	D3	D4
	1	0	0	0
1	0	1	0	0
2	0	0	1	0
3	1	0	0	1
4	1	1	0	0
5	0	1	1	0
6	1	0	1	1
7	0	1	0	1
8	1	0	1	0
9	1	1	0	1
10	1	1	1	0
11	1	1	1	1
12	0	1	1	1
13	0	0	1	1
14	0	0	0	1
15	1	0	0	0

Output sequence

Initially the flipflops D1 to D4 hold the value 1 0 0 0 respectively.

After 1st clock pulse, the data in D1 is shifted to D2, the data in D2 to is shifted to D3, the data in D3 to is shifted to D4. The data from D4 and D3 is XORed (or modulo 2 addition is performed) is fed back to the flipflop D1.

The PN sequence is take out from the flipflop D4.

After N = 15 the flipflops hold the data exactly same as the initial state and the sequence repeats.

Algorithm:

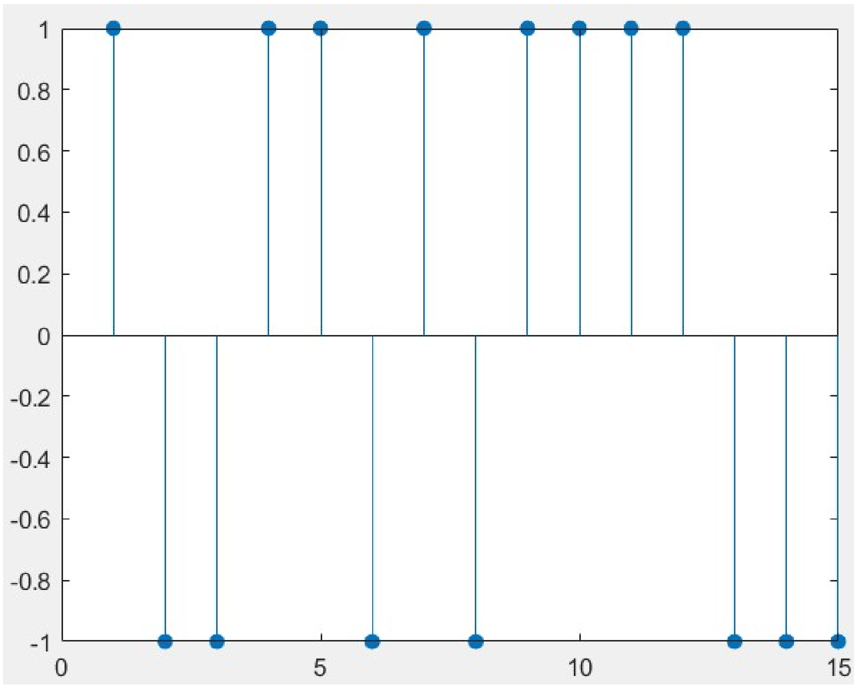
1. Read the initial state.
2. Read the generator polynomial.
3. Calculate the number of flipflops from the length of the initial state.
4. Calculate the period of the PN sequence.
5. Using looping structure go on shifting the data and also perform modulo 2 additions on the data specified by the generator polynomial coefficients.
6. Store the output of the sequence and display the same.
7. Encode the sequence +1 for bit 1 and -1 for bit 0 and plot the same.

MATLAB Script:

```
clear all;close all; clc;
init_state = [1 0 0 1];% Initial state of the shift register
M = length(init_state);% Number of shift register
N = (2^M)-1;% Length of the PN sequence
out_seq = [];% Variable to hold the output
temp1 = init_state;
for i=1:N
    out_seq = [out_seq,temp1(1)];
    temp2 = circshift(temp1,-1);%shifting of data
    % implementation of generator polynomial
    % x^4+x^3+1:
    temp2(end) = xor(temp1(1),temp1(2));
    temp1 = temp2;
end
disp('The PN sequence for the generator polynomial 43:');
disp(out_seq);
out_seq(out_seq==0)=-1;
figure(1),stem(out_seq,'filled');
```

Output:

```
The PN sequence for the generator polynomial 43:
     1     0     0     1     1     0     1     0     1     1     1     1     0     0     0
```



Note: Students have to verify the following properties of the PN sequence generated:

(i) Run Property
(ii) Balance Property
(iii) Autocorrelation Property

Inference:

Evaluation:

Attendance (2)	
Journal (3)	
Conduction (5)	
Viva (5)	
Total (15)	
Signature of faculty-in-charge with date	